RoboCup2005
Rescue Robot League Competition
Osaka, Japan
July 13 – 19, 2005
www.robocup2005.org

RoboCupRescue - Robot League Team ALCOR, Italy

Andrea Carbone[1], Giorgio Ugazio[1], Alberto Finzi[1], Fiora Pirri[1],
Andrea Orlandini[2], Marta Cialdea[2]

[1] ALCOR Laboratory
Dipartimento di Informatica e Sistemistica
Università di Roma "La Sapienza"
Via Salaria 113, I-00198 Roma, Italy
{carbone, ugazio, finzi, pirri}@dis.uniroma1.it
http://www.dis.uniroma1.it/~alcor
http://www.aalcor.org/

[2] DIA
Dipartimento di Informatica e Automazione
Università degli Studi di Roma Tre
Via della Vasca Navale 79, I-00146 Rome, Italy
{cialdea,iarusso,orlandin}@dia.uniroma3.it
http://web.dia.uniroma3.it/

Abstract. We present a stable and autonomous system based on a strong interaction between visual cognitive inference and actions executed. Mapping, Vision and Navigation are all collaborative agents that work together sharing data to accomplish to complex identification and mapping tasks. The visual process reacts to interesting features in the arena trying to approach the focused area in order to achieve more successful observations from the environment. This activity is performed coordinating the states that model the robot actions in the arena.

Introduction

We present in this competition an approach to cognitive robotics architecture to detect and identify safety issues in indoor environment. Our approach is based on a good and stable mapping and navigation system, built upon a probabilistic model of sensors data fusion, and a fuzzy controller suitably mixing several behaviors adapting to the situation. Given this basic structure we are experimenting with a complex visual cognitive system able to elaborate over some specific models of texture processing (e.g. wavelet transforms), primitive shape components, and localization of both particular moving features, and unusual people pose and gestures. This kind of visual inference concerns e.g. someone lying on the ground, or targeted gestures for signaling request for help. In the rescue arena scenario this approach leads the robot to

behave like a remote and autonomous rescue operator. The main tasks that Doro follows are: exploration of the arena and victim localization. As there is no a-priori initial knowledge about the arena map, the first task is accomplished by giving Doro a 'greedy' behavior versus unexplored regions in the map. This means that Doro is continuously attracted towards zones which have not been visited yet. This task involves activities such: wall-following, door crossing, obstacle avoidance and target approaching. The second task is carried out thanks to Doro's main sensory device: the camera. While moving, Doro registers every significant pattern or texture that may lead to a victim's identification. When a feature, or set of features, identifying a victim, are detected, the main inference system suggest to the navigation module a direction to follow in order to verify if such hypothesis leads to a clear victim's identification.

In short: the robot can be seen as a smart agent driving the attention to promising location in which to discover a victim. When such signs are not perceived, Doro continues his exploration inside the arena building the metric and topological map. All layers involved in this process (mapping, visual perception, navigation) are seen as collaborative agents more than a strict Master/Slave relationship. The visual cognitive system affects motion, the navigation system has access to the map to decide feasible path to target location, and the mapping module enrich the topological map with landmarks based on visual perception. Every participant agent shares a common area in which various tasks and goals are performed and information exchanged, in order to let the information flow run through all layers in a transparent fashion.

Our Robot Team is now enriched with a new arrival. The Shrimp III is an advanced solution for mobile robotics for rough terrain produced by the Bluebotics SA (http://www.bluebotics.com), a spin-off company of the Autonomous Systems Lab, EPFL (Lousanne).Its kinematic design allows the climbing of unstructured and rough grounds (typical of the yellow and red arenas). At the time of the writing this document we are involved in the equipment of the raw mobile base in order to make it suitable for the rescue competitions that will be held in Osaka (remote control by an operator, cams and so on). Being especially designed to face hostile enviroments that our robot DORO cannot negotiate, we plan to take advantage of its capabilities to challenge more difficult arenas.

1. Team Members and Their Contributions

- Fiora Pirri — Visual Cognitive System.
- Andrea Carbone — Main Operator – Vision algorithms, Navigation, Software Design.
- Giorgio Ugazio — SLAM, HRI, Path Planning, Software Design
- Alberto Finzi — Golog Planner.
- Andrea Orlandini — Golog Planner
- Marta Cialdea — Golog Planner

2. Operator Station Set-up and Break-Down (10 minutes)

Doro set-up requires few steps to be performed. They consist on: turning on of both the operator and remote laptops followed by the starting of the onboard Doro Platform Server (it could be done via remote desktop). These steps allow each other module (including the operator) to gain control permission over the Robot and to start the exploration of the targeted arena. On top of the robotic platform we placed various devices (cameras, pan-tilt and telemeter). They are all connected to the on- board

laptop. All is kept together by a plastic support which contains the laptop and all requested cables and power supply adapters. ActivMedia Pioneer robot is provided with an external handle on the rear in order to let the operator to bring it up and transport like a bag.

3. Communications

The laptop mounted on the top deck of Doro is equipped with an **802.11A** wireless PCMCIA device. This provides access to all the on-board activities via a remote monitoring using an operator laptop which is connected in a peer-to-peer fashion with the Doro laptop. We plan to call this dedicated network: "**DORONET**". We are going to use standard C class 192.168.10.x addresses.

4. Control Method and Human-Robot Interface

In this section we describe our Communication System, Control System, Interaction System and our operational modes on both Doro and Shrimp robots.

- **Doro Platform**

To minimize the amount of work to be done by on-board Doro laptop we have built a Doro Platform who has the following responsibility:

- **Ensure a robust connection to the robot hardware**. This task is accomplished by active clients that periodically check the presence of the connections (to the Robot, o the PTU, to the Laser, etc.) and in case of a communication breakdown tries to re-establish them.

- **Handles Robot access Permission.** High-Level modules need different access permission to the Robot. For example a SLAM System needs to adjust robot position but should not attempt to directly modify the current action of the robot itself. A kind of privilege granted only to the Navigation Subsystem. A GUI should allow operators to do everything, while a Viewer (see further) only needs to 'observe' the robot state without ever changing it. Doro Platform handles robot access permission. There are 4 types of permissions, in order of importance: **Viewer** (to just receive Doro Data), **Localizer** (to modify Robot Position), **Navigator** (to gain access to Robot Motors), **Administrator** (to control the overall Robot System, including other module permissions).

- **Handles Robot Localized Coordinate System.** ActivMedia Robotics Interfaces for Application (ARIA) libraries allow programmers to gain access to the internal odometric robot estimated pose. This functionality is very useful, for example, when a module with Localizer Permission (Slam, Operator, etc.) needs to adjust Robot Pose. This task is accomplished by building an own over-structured Coor-

dinate System that takes care of Doro internal odometric pose and Localizer actions.

- **Handles Robot Motion.** ARIA allows programmers to implement behaviors in term of action groups. An Action group requires a Local Robot Handler so cannot be remotized and needs to run on local machine. This is the reason why we've embedded robot motion control inside the Platform. High-Level modules with a Navigator Permission (such Navigation System, Operator, etc.) can activate whatever actions they desire to combine them in a group. Elementary actions include Joystick driving, so operators can have robot access using the same interface of Navigation System.
- **Handles Robot Data.** All interesting data from the robot are collected in a high-level class. Doro Platform continuously updates these data (internal position, estimated position, sonar data, battery level, etc.) and dispatches them to Doro Platform Clients with a Viewer Permission.

In short: Doro Platform wraps up both the hardware and software modules handling at the same time the policies for their management.

• Doro Heart

In our design high-level modules don't communicate directly with the Platform. The backbone of the whole system is Doro Heart: a Central Communicating System composed by:

- **State Manager:** a service that holds useful shared information about every high-level module. For example a Vision module can share victims position and a Viewer module can draw the victims on a window. States are updated by each module through a State Updater Server and read through a State Monitor.
- **Task Manager:** a service that dispatch tasks to each module. Tasks are sent by Planner (in Autonomous Mode) using a Task Dispatcher and are received by each module using the Task Receiver. When not in Autonomous Mode, some tasks are assigned by the Operator and the Planner itself has to listen to its task as a common module. In Totally Teleop Mode the Operator has a total control over each module and Planner runs in idle.
- **Platform Client:** as previously mentioned, high-level modules don't communicate directly with the Platform. Instead a unique Platform Client (with all permission) is shared between modules. Modules can access the shared Platform Client using their shared state information. Access control is filtered by Doro Heart.

Each module has a Heart Client (HC) that contains a State Monitor (to watch other states), a State Updater (to update own state to the State Manager) and a Task Receiver (to listen to the task given by a Planner or Operator).

• Doro Head

So far we've spoken about Doro embedded hardware (Sonar, encoders, motors,…), now we are going to speak about what we call Doro Head: a pair of CCD Cameras and a Laser Telemeter mounted on a Pan/Tilt Unit. Doro Head is simply the collection of software services that handles this hardware.

- **PTU Service:** a Server that allow users (Slam System, Vision System, Laser Service, Operator) to take control of the Pan/Tilt Unit. In this way it's possible to reset PTU position, set a pair of angle (Pan, Tilt) and force PTU to reach them or just read the actual PTU position.
- **Laser Service:** a Server that allows users (Slam System, Vision System, and Operator) to take a single measurement or a Laser Scan (a set of measurements on ±90 degrees from robot direction). Laser Service, while used as a scanner, needs to be a Client of PTU Service. Conflicts are resolved by the Planner.
- **CCD Cameras:** the only massive communication from Doro Laptop and Operator Laptop is Cameras data stream. At the moment of TDP building we're trying to find an optimal solution to Cameras data transferring. Our decision must satisfy the following requirement: (i) Image Frame should be acquired in the same environment where they are going to be elaborated; (ii) Transmission should be fast and should be minimized LAN occupancy; (iii) Frames should be at least of 320x240 pixels and Frame Rate at least 10 FPS; (iv) Camera Images should be embedded inside the GUI (that is written in C++). We're at work!

In few words, Doro Head is just an abstraction that incorporates all Head Features.

• GUI (AHRIS)

AHRIS stays for Alcor Human Robot Interface System and it's a **GUI** (Graphic User Interface). with AHRIS an Operator can control the Robot or just make a supervision of the whole robotic system. Up to now AHRIS doesn't embed video streaming (see Doro Head section for further information).

• Operational Modes

Doro has several components. Operator can requires control over none, part of all of them and, consequently, Planner module releases the required resources. Some modules (as Slam System, Navigation System and Vision System) can also be partially controlled by the Operator so the overall controllability is maximized. In that way we realize a gradual variation from Totally Teleop to Totally Autonomous that we cluster in a common definition: Supervised.

• Control System

A model-based executive control system supervises and integrates both robots modules activities and the operator interventions. Main robot and operator processes are explicitly represented by a declarative temporal model which permits a global interpretation of the execution context. Given this model, a reactive planner can monitor the system status and generate the control on the fly continuosly performing sense-plan-act cycles. At each cycle the planner is to: generate the robot activities up to a planning horizon, and monitor the consistency of the running activities managing failures. In this setting, the human operator can interact with the control system influencing the planning activity in a mixed initiative manner ([5]).
The reactive planner was developed following an high level programming approach, deploying the Temporal Concurrent Golog [6]: temporally flexible plans are pro-

duced by a Golog interpreter which completes partially specified behaviour (Golog programs) selected from a plan library.
The main DORO processes and states are explicitly represented by a declarative dynamic-temporal model specified in the Temporal Concurrent Situation Calculus [6]. The system is modeled as a set of components whose state changes over time (analogously to Constraints Based Interval Planning paradigm [7]).
For each execution cycle, once the status is updated (sensing phase), the Golog interpreter (planning phase) is called to extend the current control sequence up to planning horizon. When some task ends or fails, new tasks are selected from the task library and compiled into flexible temporal plans filling the timelines.
Under nominal control, the robot's activities are scheduled according to a closed-loop similar to the LOVR (Localize, Observe general surroundings, look specially for Victims, Report) sequence in [4].

5. Map generation/printing

As usual we conceive two levels of mapping: the Global Map (GM) and the Local Map (LM), the latter is obtained at each SLAM-cycle, via sonar sensor fusion; the LM is incrementally integrated into the GM, exploiting the well known Bayesian approach [1, 2]. Integration follows a correct alignment of the robot with the map [2], suitably reviewing odometric errors. To make the map worth for the operator, so that he can follow the path pursued so far, and verify whether some possible thread has not yet been visited, we introduce two concepts the History Path (HP) and the Exploration Path (EP), the first (see the blue ribbon in Figure 5.1) displays the whole path threaded from the very beginning, and the latter (look at the yellow ribbon) makes evident the last steps taken. A typical scenario where the paths are useful is when it is necessary to draw conclusions like "this area has already been visited", "he victim currently observed is inaccessible from this path, take a snapshot and try to reach it from another way", and so on. High-Level Software Module that accomplishes this job is Doro SlamSy:

- **DORO SLAMSY**

Doro Slam System (SlamSy) tasks are various: from mapping and localizing to path planning. Each task is accomplished by a sub-module we call Agent. Here we describe agents:

- **EETA:** End of Exploration Test Agent. It's an agent whose only goal is to determine when Exploration Phase is over. This decision is made in functions of some factors: (1) Time elapsed since exploration start; (2) Area explored during current exploration; (3) Exploration Path length. We define a function **W(t)** that weightily sums these factors and when that function became greater then a threshold value τ then Exploration phase is over. Reaching the threshold is guaranteed because each factor is monotonic (especially Time Elapsed!).
- **NUFA:** Nearest Unexplored Finder Agent. While $W(t)<\tau$ the Nearest Unexplored region is evaluated assigning a score to each cell of the current GM, and then by

an optimized version of Value Iteration Algorithm [3] the minimum cost path (Unexplored Path or UP) toward the unexplored cell is found. An indication of the nearest unexplored (NU) zone appears on the panel as an arrow pointing towards this direction (see the dark red arrow on the map, in Figure 5.1(up)) and another ribbon (the red one) indicates the UP. Our optimized version of the Value Iteration Algorithm is based on updating on-line cell values using yet-updated neighbor's value. So value propagation gently follows cell index cycle. To avoid counter-current value propagation (when a cell value needs to be updated using cells that has to be scanned) map cells are cycled from all eight planar wind direction and only a rectangular box, containing all modified value and their neighbors, is updated each cycle. In this way we've seen that from a theoretically complexity of $o(n^2)$, where "n" are the cell number (a measure of explored surface), we've reached a practical complexity of o(n) because the number of the updating cycle over the whole map has never been more than eight (one for each direction) up to now.

- **VPFA:** View Point Finder Agent. On the other hand, when $\mathbf{W(t)>\tau}$, the exploration is over and then the search looks for an optimal vantage point, i.e. the spot to reach and the direction of sight. We'll call this position View Point (VP). The View Point is obtained maximizing a functional **J(s)** defined over the map cell of the last explored area (the map cells that has changed occupancy value in last exploration phase). J(s) weightily sums four components: (1) **Distance from the gravity center**: the VP is optimal if, given the other conditions, its distance from the center of gravity of the area explored during so far, is minimal (motivation: robot shouldn't forget to analyze the whole explored area); (2) **Distances in a view path**: the VP is optimal if, given the other conditions, its distance from each point in the view path (i.e. the set of VP found previously) is maximal (motivation: robot shouldn't lose time to watch where it has yet watched); (3) **Distance from the current position**: the VP is optimal if, given the other conditions, its distance from the current robot position is minimal (motivation: robot shouldn't spend so much energy and time to go to the VP); (4) **Quality**: the VP is optimal if, given the other conditions, the area to which it belongs is free from obstacles and with a wide visual angle (motivation: robot shouldn't place itself in a rough or occluded area). The weighting factors can be supposed constant for this simple analysis. One of our goals is to make them adaptive in function of some environmental property or learned. Maximization is obtained by Hill Climbing, with random restart.
- **PFA:** Path Finder Agent. After calculating the VP, robot first objective is to reach it. So, while it has not yet reached its goal, the View Point Path (VPP) is calculated using the same technique for calculating the UP: Optimized Value Iteration. In figure 5.1(up) you can see the black arrow that indicates View Point and the red ribbon that indicates View Point Path. In both cases (View Point and Nearest Unexplored) a partial goal, called Desired Position (DP), is calculated (the light red arrow in figure 5.1) to drive the robot through the right position.

So Slam System is a complex thing that *also* makes the map! Map can be saved in common format (such as JPG or PNG) and contains the Occupancy Grid, Metric

Information, Robot Path, Victims found and other things useful for human rescue operator. Additional information will be added if considered useful by the operator directly using the client interface.

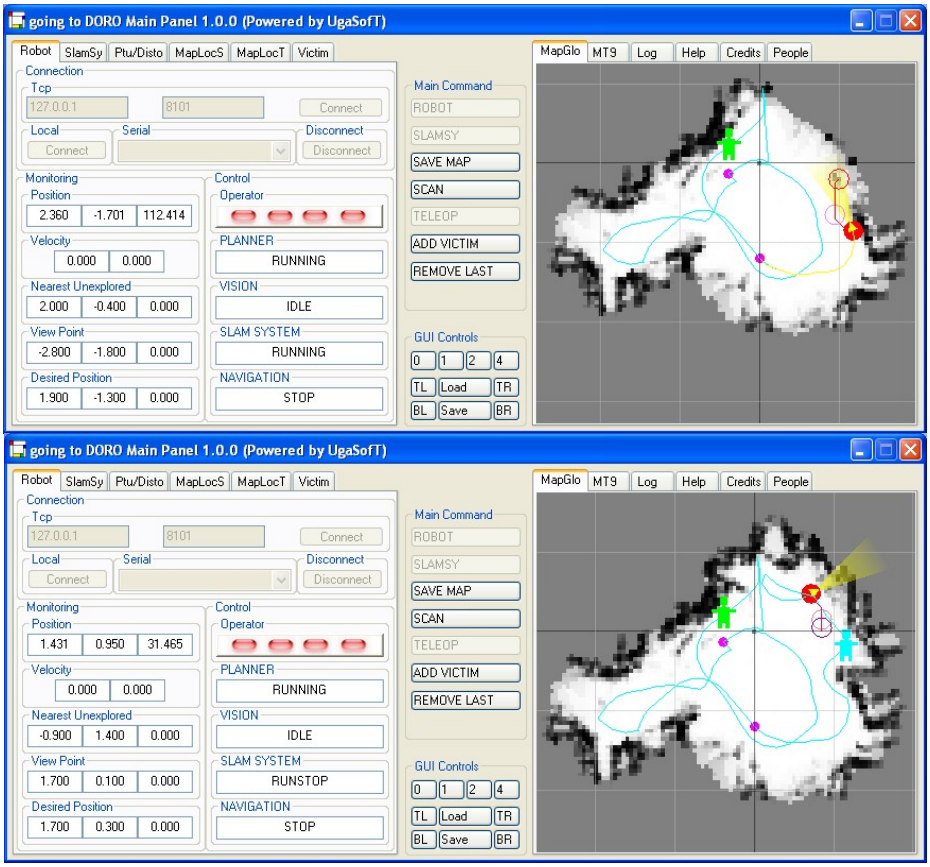

Figure 5.1: Snapshot of our current GUI.
up: NUFA in Actions
down: VPFA in actions

6. Sensors for Navigation and Localization

The pioneer is equipped with 8 Sonar ranging 180° in forward direction. A telemeter is mounted on the same pan-tilt unit that hosts the camera. Sensor data are fused and mapped into local occupancy value. Local occupancy value must be integrated in time to build a single, global, Metric Map using a Bayesian Filter. A Bayesian Filter is a recursive estimator we use to calculate sequences of posterior probability distribution over a quantity that cannot be observed directly: the Map. The final result is a set (a matrix) of cells each representing the probability of being occupied. The sonars

produce distances with a short period, so they provide a continuous flow of information to the mapping routines. The telemeter, being joined to the same pan-tilt unit that supports the camera, requires a custom task that provides high-confidence distances when the robot enters a new region. In this domain with high-confidence measure we mean a sensor data that strongly raise the occupancy value of the spatial cell it represents. The localization issues are solved with a fast integration routine of the Inertial Platform data fused with odometric data coming from the wheels.

7. Sensors for Victim Identification

The main sensor devoted to victim identification is the camera. The visual inference stages is composed by different perceptual stages which goes from an Attention phase in which an interesting 'feature' in the environment catches the attention of the agent, to a Cognitive inference based on further (and appropriate) observations are taken into account to guess if all perceptual data collected can lead to a success in the identification of a specified target. The visual process is carried by mean of dedicated image analysis routines that can be grouped in more stages like: segmentation/clustering, edge finding, features extraction (wavelets) and others.
To detect victim sounds (speech or just finger tapping) we have mounted in front of Doro a directional microphone and a radio transmitter. Our experience in last RoboCupRescue (Lisboa) with sound detection was very good but during the final we fell in a False-Positive due to unnatural noises.

8. Robot Locomotion

Doro is a two-wheeled differential drive robot with a caster on the rear. No particular modifications have been done to the original ActivMedia design for what concerns the locomotion.

9. Team Training for Operation (Human Factors)

The user interface gives the operator a complete control over the robot. So just a basic training in an artificial arena will be useful to take confidence with the controls. Our aim is to build an autonomous system, so the only effort requested to an operator is to familiarize with all the signals and controls provided by the user interface.

10. Possibility for Practical Application to Real Disaster Site

Being based mostly on vision, this system can be highly useful to bring the attention of a remote operator toward specified location of an area that the visual inference system evaluates as a candidate for a rescue action. Basing on the same process the developed visual inference can be applied to a wide range of task that can help a rescue operator to gain informations about the operative condition in a disaster area.

11. System Cost

The economic effort requested by the devices ranges from 8000€ to 10000€. The main cost can be accounted for the mobile robot (half of the overall cost). The entire set is composed by:

- ActivMedia PioneerDX
- Laptop Asus centrino (1.6)
- ImageInAction framegrabber (connects to 1394 firewire)
- Direct Perception PTU
- Xsens MT9 Inertial Platform
- Sony XC99 CCD camera

References

1. S. Thrun, “Robotic mapping: A survey,” in Exploring Artificial Intelligence in the New Millennium, G. Lakemeyer and B. Nebel, Eds. Morgan Kaufmann, 2002.
2. S. Thrun, Learning Metric-Topological Maps for Indoor Mobile Robot Navigation. Artificial Intelligence, 99(1):21-71, 1999.
3. R. Bellman, Dynamic Programming. Princeton: Princeton University Press, 1957.
4. R. Murphy, “Human-robot interaction in rescue robotics,” IEEE Transactions on Systems, Man and Cybernetics, Part C, vol. 34, no. 2, pp. 138–153, 2004.
5. M. Ai-Chang, J. Bresina, L. Charest, A. Chase, J.-J. Hsu, A. Jonsson, B. Kanefsky, P. Morris, K. Rajan, J. Yglesias, B. Chafin, W. Dias, and P. Maldague, “Mapgen: mixed-initiative planning and scheduling for the mars exploration rover mission,” Intelligent Systems, IEEE, vol. 19, no. 1, pp. 8– 12, 2004.
6. R. Reiter, Knowledge in action : logical foundations for specifying and implementing dynamical systems. MIT Press, 2001.
7. A. K. Jonsson, P. H. Morris, N. Muscettola, K. Rajan, and B. D. Smith, “Planning in interplanetary space: Theory and practice,” in Artificial Intelligence Planning Systems, 2000, pp. 177–186.